

# Adding Query Privacy to Robust DHTs

Michael Backes<sup>1,2</sup>   Ian Goldberg<sup>3</sup>   Aniket Kate<sup>1</sup>   Tomas Toft<sup>4</sup>

<sup>1</sup>Max Planck Institute for Software Systems (MPI-SWS), Germany

<sup>2</sup>Saarland University, Germany

<sup>3</sup>University of Waterloo, Canada

<sup>4</sup>Aarhus University, Denmark

## Abstract

Interest in anonymous communication over distributed hash tables (DHTs) has increased in recent years. However, almost all known solutions solely aim at achieving sender or requestor anonymity in DHT queries. In many application scenarios, it is crucial that the queried key remains secret from intermediate peers that (help to) route the queries towards their destinations. In this paper, we satisfy this requirement by presenting an approach for providing privacy for the keys in DHT queries.

We use the concept of oblivious transfer (OT) in communication over DHTs to preserve query privacy without compromising spam resistance. Although our OT-based approach can work over any DHT, we concentrate on communication over *robust* DHTs that can tolerate Byzantine faults and resist spam. We choose the best-known robust DHT construction, and employ an efficient OT protocol well-suited for achieving our goal of obtaining query privacy over robust DHTs. Finally, we compare the performance of our privacy-preserving protocols with their more privacy-invasive counterparts. We observe that there is no increase in the message complexity and only a small overhead in the computational complexity.

*Keywords:* Distributed hash tables, Query privacy, Spam resistance, Oblivious transfer

## 1 Introduction

In the digital society, our online activities are persistently recorded, aggregated, and analyzed. Although worldwide electronic data privacy laws and organizations such as EFF [1] and EPIC [2] try to challenge this pervasive surveillance through policies and protests, privacy enhancing technologies (PETs) are key components for establishing a suitable privacy protection mechanism from the technology side. The interest in developing novel PETs is increasing for a variety of reasons, ranging from the desire to share and access copyrighted information without revealing one’s network identity, to scalable anonymous web browsing [29–31, 35, 50]. In this paper, we study privacy in the peer-to-peer (P2P) paradigm, a popular approach to providing large-scale decentralized services.

In the P2P paradigm, distributed hash tables (DHTs) [38, 41, 47, 54] are the most common methodology for implementing structured routing. Similar to hash tables, a DHT is a data structure that efficiently maps *keys* onto *values* that are stored over a distributed overlay network. However, unlike hash tables, DHTs can scale to extremely large number of key-value pairs as the mapping from keys to values is distributed among all peers. In order to obtain a value associated with a key, a requester (a sender) in a DHT routes the key through a small fraction of the network to reach the receiver that has stored the value. DHTs can also handle continual arrivals and departures of peers, and small-scale modifications to the set of peers do not disturb the mapping from keys to values significantly.

In a DHT, privacy may be expected for the sender, the receiver or the queried key. Ensuring the anonymity of senders and requesters in DHTs has received considerable attention in the privacy commu-

nity [29–31, 35, 50]. Privacy of the queries / keys, i.e., keeping the keys secret from intermediate peers that route the queries towards their destinations, is also equally important: in many scenarios such as censorship resistance, this query privacy constitutes a necessary condition for sender and requestor anonymity. In this paper, we present a practical approach to obtain privacy for queries in *robust and spam-resistant* DHTs where a fraction of peers may behave maliciously.

## 1.1 Contributions

Almost all anonymity solutions for DHTs try to provide anonymity to a sender or a requester in a DHT lookup, upload or request. It may also be necessary that the queried key remains secret from peers that route the corresponding requests in some situations. We call this property *query privacy*. In this case, an intermediate routing peer should be able to suggest a next peer or a set of next peers without determining the key being searched for. Example application scenarios for this property can be protection against mass surveillance or censorship, preventing tracking and data-mining activities on users requests, and providing opportunities to access material that is socially deplored, embarrassing or problematic in society.

*Recursive* routing and *iterative* routing are the two approaches to route information in DHTs. In the *recursive* routing approach, obtaining query privacy looks infeasible, if not impossible. This results from the fact that the intermediate router itself decides to which peer to forward a request. Assuming that every peer is under the control of an individual, it is always possible for the controller to figure out the next peer for the request. On the other hand, query privacy in *iterative* routing, which is also a commonly used routing approach over robust DHTs, can be trivially obtained if every peer sends its complete routing table to the requesting peer. The requester can then determine the next peer itself and send the request. However, this solution may make it significantly easier to mount spamming attacks in the systems: a malicious sending peer can easily gather a significant amount of routing information, and use it to determine and target peers that hold specific keys.

We instead use the oblivious transfer (OT) primitive [17,37]. Given a peer holding a database, OT allows a requester with a key to obtain a database entry associated with the key, such that the requester does not get any information about other database entries and the database owner does not learn the requester’s key. Therefore, OT perfectly fits our requirements of obtaining query privacy without divulging additional routing information. We use the OT protocol by Naor and Pinkas [32] in the best-known robust DHT constructions by Young et al. [52, 53] (their RCP-I and RCP-II protocols) to obtain our goal of query privacy in robust DHTs. Importantly, our query privacy mechanism does not increase the message complexity of the RCP-I and RCP-II protocols and an increase in the computation cost is also not significant. We elaborate on our exact choice of OT protocol in Section 4.3, and discuss robust DHT constructions in Section 2.

The employed OT protocol [32] is a simple indexed OT protocol, where the database contains index-value pairs and a requester inputs an index. However, a query for a routing table entry is an interval membership (or a range) query and not an index query. Therefore, to prevent a requesting peer from obtaining *any* additional information, we could have employed the concept of conditional OT (COT) [15] and used the interval-membership strong COT (I-SCOT) protocol by Blake and Kolesnikov [8]. However, the I-SCOT protocol is expensive in terms of both computation and communication. We observe that by releasing the upper and lower bounds of a routing table entirely to the requester, large improvements can be made. There is no information in these range boundary values for a malicious requesters in terms of spamming as they do not convey any information regarding the identities of the key owners. However, given this information, the requester *will know* the desired entry (index), allowing the use of OT instead of the more complex COT.

Private information retrieval (PIR) [13] is a weaker form of OT, where more information may be revealed than asked for; e.g., sending the complete routing table is a trivial PIR protocol. PIR protocols can be less costly than OT in terms of computation, but the risk of spamming persists with all non-OT PIR protocols, and hence we avoid them.

**Outline** The rest of the paper is organized as follows. In Section 2, we survey the literature on robust DHTs. Section 3 describes our system model, while Section 4 overviews the cryptographic tools used in our constructions. In Section 5, we present the robust communication protocols that preserve query privacy. In Section 6, we analyze and discuss performance and systems issues. Finally, we conclude in Section 7. We include a detailed description of the employed OT protocol in Appendix A.

## 2 Background and Related Work

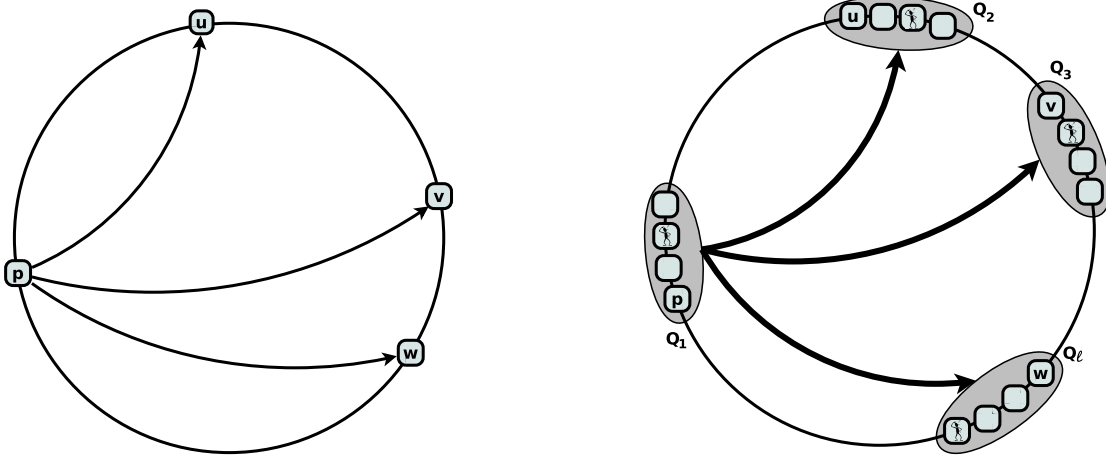
Malicious behaviour is now common over the Internet. Lack of admission control mechanisms in DHT systems make them particularly vulnerable to these malicious (or Byzantine) attacks [45, 49]. Such attacks can not only pollute the data that is available over DHTs [25], but also poison the indices by creating fake data identifiers [26]. They may further create Sybil identities and disrupt communication between well-behaving peers by spamming. The concern is quite serious, since large-scale P2P systems in existence today (e.g., Azureus or Vuze DHT [18] or KAD DHT [46]) see millions of users every day. Along with the basic file sharing application, there are proposals for using P2P systems to re-implement the Domain Name System [48], mitigate the impact of computer worms [3] and protect archived data [20, 51]. These applications would benefit from tolerance against Byzantine behaviors. As a result, a number of solutions have been defined that can provably tolerate Byzantine faults over P2P systems (e.g., [5–7, 19, 22, 23, 33, 42, 52]). Due to the popularity of DHTs, the majority of these solutions are built to work over DHTs and the resulting constructions are called *robust* DHTs.

In robust DHTs, malicious attacks are generally dealt with using the concept of *quorums* [4–7, 19, 22, 33, 42]. A quorum is a set of peers such that a minority of the members suffer adversarial faults. Typically, it consists of  $\Theta(\log n)$  nodes where  $n$  is the total number of nodes in the underlying DHT. A DHT quorum replaces an individual DHT peer as the atomic unit and malicious behaviour by an adversary is overcome by majority action; e.g., the content may be stored in a distributed and redundant fashion across members of a quorum such that it cannot be polluted by a small fraction of host peers. Poisoning attacks can be mitigated by having peers belonging to the same quorum validate routing information before it is advertised. If a peer violates the protocol, then it is possible to remove it from the quorum, which effectively removes them from the system.

Protocols using quorums are split between those that use iterative and those that use recursive approaches. When sending a request using the recursive approach, a sending peer contributes one message (its request), while its DHT has to generate  $O(\log n)$  messages. In the iterative approach, a sending peer has to contribute an equal number of message as its DHT. Consequently, while dealing with Byzantine faults, the iterative approach is more common than the recursive approach as the former provides better protection against the spamming attack than the latter.

The common way such quorums are utilized is as follows: a request  $m$  originating from a peer  $p$  traverses a sequence of quorums  $Q_1, Q_2, \dots, Q_\ell$  until a destination peer is reached. A typical example is a query for content where the destination is a peer  $q$  holding a data item. Initially  $p$  notifies its own quorum  $Q_1$  that it wishes to transmit  $m$ . Each peer in  $Q_1$  forwards  $m$  to all peers in  $Q_2$ . Every peer in  $Q_2$  determines the correct message by majority filtering on all incoming messages and, in turn, forwards it to all peers in the next quorum. This forwarding process continues until the quorum  $Q_\ell$  holding  $q$  is reached. Assuming a majority of correct peers in each quorum, transmission of  $m$  is guaranteed.

Unfortunately, this simple protocol is costly. If all quorums have size  $\eta$  and the path length is  $\ell$ , then the message complexity is  $\ell\eta^2$ . Typically, for a DHT of  $n$  nodes,  $\eta = \Theta(\log n)$  and, as in Chord [47],  $\ell = O(\log n)$ , which gives  $O(\log^3 n)$  messages; this is likely too costly for practical values of  $n$ . Saia and Young [42] mitigate this problem using a randomized protocol which provably achieves  $O(\log^2 n)$  messages in expectation; however, the constants in their protocols are prohibitively large.



Peers  $u$ ,  $v$ , and  $w$  are in the routing table ( $\mathcal{RT}$ ) of peer  $p$  on a DHT. Correspondingly, in an quorum topology with  $p \in Q_1$ ,  $u \in Q_2$ ,  $v \in Q_3$  and  $w \in Q_\ell$ , quorums  $Q_2$ ,  $Q_3$ , and  $Q_\ell$  are linked to quorum  $Q_1$  in its  $\mathcal{RT}$ . Thick lines signify inter-quorum links. Each quorum has size  $\eta = \Omega(\log n)$  and must have strictly fewer than  $1/3$  faulty peers.

Figure 1: Quorum Topology over DHTs

Recently, Young et al. [52] demonstrated that the problem can be solved using threshold cryptography [14]. Using a distributed key generation (DKG) protocol over the Internet [24] and a threshold digital signature scheme [9], they design two robust communication protocols, RCP-I and RCP-II, that respectively require  $O(\log^2 n)$  messages and  $O(\log n)$  messages in expectation. Importantly, these protocols can tolerate adversarial peers up to any number less than  $1/3$  of a quorum in the asynchronous communication setting and less than  $1/2$  of a quorum in the synchronous communication setting. They also do not require any trusted party or costly global updating of public/private keys outside of each quorum. The protocols work in the elliptic curve cryptography (ECC) based discrete logarithm setting, and its security is based the gap Diffie–Hellman (GDH) assumption [11]. The paper also includes results from microbenchmarks conducted over PlanetLab showing that these protocols are practical for deployment under significant levels of churn and adversarial behaviour. We find this work to be the most up-to-date solution for robust and spamming-resistant communication in DHTs and use it as a starting point towards query privacy.

Privacy in communication over DHTs has also been under consideration over the last few years [29–31, 34, 35, 50]. However, most of these PETs concentrate on sender (or requester) privacy, and generally aim at a scalable anonymous web browsing system: a future replacement for Tor [16]. Our aim in this paper is different; we want to achieve privacy for keys in DHT queries (or query privacy). Nevertheless, we observe that our query privacy mechanism can further enhance anonymity in almost all of the above PETs. Our approach is also significantly better in terms of message complexity than redundant routing [34], where a requester makes multiple queries to confuse an observer.

### 3 System Model and Assumptions

In this section, we discuss the quorum-based DHT system model, and the adversary and communication assumptions that we make in our protocols. As we develop our anonymity solution on top of robust communication protocols by Young et al. [52], our model is nearly the same as their model.

For ease of exposition, we do not consider the link failures and crash-recovery mechanism in that work, which in turn follows from the underlying DKG architecture [24]. However, our protocols indeed work even under these assumptions without any modification.

### 3.1 Adversary and Communication Assumptions

We work in the asynchronous communication model (unbounded message delays) with Byzantine faults. However, to ensure the liveness of the protocols, we need the *weak synchrony* communication assumption by Castro and Liskov [12], which states that the message delay does not grow longer indefinitely. Note that this assumption arrives from the underlying robust communication protocols and is unrelated to our privacy preserving mechanism.

In a P2P system, each peer is assumed to have a unique name or identifier  $p$  and an IP address  $p_{\text{addr}}$ . Peers  $p$  and  $q$  can communicate directly if each has the other in its routing table ( $\mathcal{RT}$ ).

Similar to the majority of anonymous communication networks [16, 29, 30, 35], we do not assume a global adversary that can control the whole network and break anonymity by observing all communication by every peer. Such an adversary seems impractical in large-scale geographically distributed DHT deployments. However, we assume that our partial adversary knows the network topology and controls a small fraction of the DHT peers. Following prior works [39, 40, 43, 52, 53], we consider around 10% of all peers to be under adversarial control. The adversary cannot observe communication at the majority of nodes; however, it may try to break query privacy, spam honest nodes, or disrupt the communication by actively attacking traffic that reaches peers under its control.

We assume that the 10% adversarially controlled nodes are spread out evenly over the DHT, and strictly less than  $1/3$  of the peers in any quorum are faulty which is the best possible resiliency in the asynchronous setting. This bound on the adversary is possible using mechanisms like the *cuckoo-rule* developed by Awerbuch and Scheideler [6], which restricts the adversary from acquiring many peers in the same quorum. Further, all faulty peers in a quorum may be under the control of a single adversary, and collude and coordinate their attacks on privacy, safety and liveness.

Finally, our adversary is computationally bounded with security parameter  $\kappa$ . We assume that it is infeasible for the adversary to solve the GDH problem [11] in an appropriate setting for signatures and the decisional Diffie–Hellman (DDH) problem [10] in another setting for OT.

### 3.2 Quorums

In a variety of approaches used to maintain quorums, one may view the setup of quorums as a graph where peers correspond to quorums and edges correspond to communication capability between quorums. This is referred to as the *quorum topology* in the literature. Figure 1 shows how quorums can be linked in a DHT such as Chord [47].

We assume the following four standard invariants [52] are true for the quorum topology under consideration:

**Goodness.** Each quorum has size  $\eta = \Omega(\log n)$  and must have strictly fewer than  $1/3$  faulty peers.

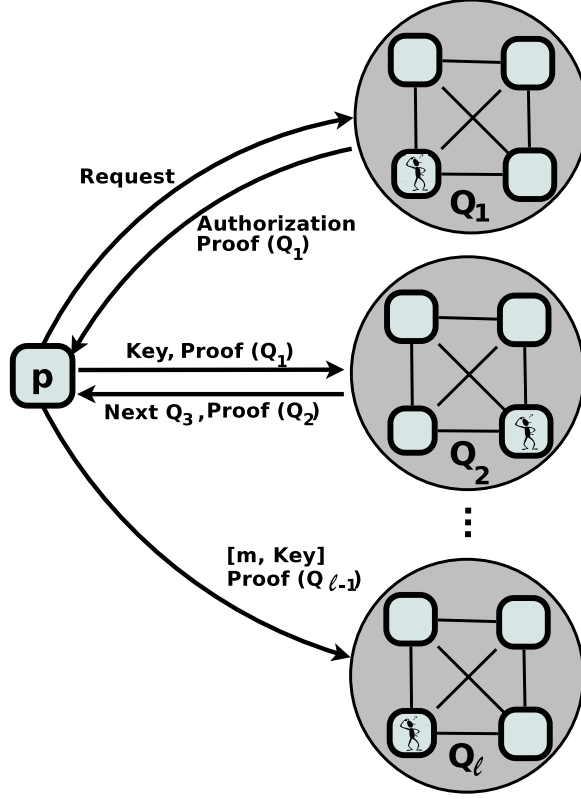
**Membership.** Every peer belongs to at least one quorum.

**Intra-Quorum Communication.** Every peer can communicate directly to all other members of its quorums.

**Inter-Quorum Communication.** if  $Q_i$  and  $Q_j$  share an edge in the quorum topology, then  $p \in Q_i$  may communicate directly with any member of  $Q_j$  and *vice versa*.

To the best of our knowledge, no practical implementation of a quorum topology yet exists. However, as indicated in the literature [5–7, 19, 33], maintaining the above four invariants looks plausible in real-world DHTs.

In a DHT where the above four invariants are maintained, the general communication mechanism in Young et al. [52] works as shown in Figure 2. Assume that a peer  $p$  wants to send a query  $m$  associated



A peer  $p$  sequentially communicates with  $Q_1$ ,  $Q_2$ , and so on, until it reaches  $Q_\ell$  who owns the searched-for key.

Figure 2: Iterative Communication in Robust DHTs using Quorums

with a **key** that belongs to quorum  $Q_\ell$ , which it does not know. The recipients of the request are generally a set of peers  $D \subseteq Q_\ell$ . Peer  $p$  requests authorization from peers in its quorum  $Q_1$ . These authorizations are based on a *rule set* [19] that defines acceptable behavior in a quorum (e.g., the number of data lookup operations a peer may execute during a predefined time period). This rule set is known to every peer within a quorum and is possibly the same across all quorums; it reduces the impact of spamming attacks. Peer  $p$  receives  $\text{PROOF}(Q_1)$  in the form of a signature if authorized. It then sends this to quorum  $Q_2$  from its routing table, which is responsible for the **key** being searched for. One or more members of  $Q_2$  verify the signature and provides  $p$  routing information and a  $\text{PROOF}(Q_2)$  for  $Q_3$ , which will convince  $Q_3$  that  $p$ 's actions are legitimate (i.e., approved by its quorum). The protocol continues until  $p$  reach  $Q_\ell$ .

As mentioned in Section 2, it possible to achieve robust communication without using any of the above cryptography. However, use of cryptography provides efficiency and reduce the message complexity by at least a linear factor. Note that we do not discuss membership update operations for quorums in this paper as they remain exactly the same as those in previous work [52,53].

## 4 Cryptographic Tools

Here, we describe the cryptographic tools that we use in our solution. In particular, we review distributed key generation, threshold signature and oblivious transfer protocols.

## 4.1 Threshold Signatures

The use of distributed key generation (DKG) and threshold signatures in our privacy preserving schemes comes from the underlying robust DHT architecture. In this architecture, threshold signatures are used to authenticate the communication between quorums. In an  $(\eta, t)$ -threshold signature scheme, a signing (private) key  $sk$  is distributed among  $\eta$  peers either by a trusted dealer (using verifiable secret sharing) or in a dealerless fashion (using DKG). Along with private key shares  $sk_i$  for each party, the distribution algorithm also generates a verification (public) key  $PK$  and the associated public key shares  $\widehat{PK}$ . To sign a message  $m$ , any subset of  $t + 1$  or more peers use their shares to generate the signature shares  $\sigma_i$ . Any party can combine these signature shares to form a message-signature pair  $\mathcal{S} = (m, \sigma) = [m]_{sk}$  that can be verified using the public key  $PK$ .

In this work, we refer to a message-signature pair  $\mathcal{S}$  as a signature. Further, it is possible to verify the individual signature shares  $\sigma_i$  using the public key shares  $\widehat{PK}$ . We assume that no computationally bounded adversary that corrupts up to  $t$  peers can forge a signature  $\mathcal{S}' = (m', \sigma')$  for a message  $m'$ . Malicious behaviour by up to  $t$  peers cannot prevent generation of a signature.

Among three known practical threshold signature schemes [9, 21, 44], Young et al. employed the threshold version [9] of the Boneh-Lynn-Shacham (BLS) signature scheme [11] for their robust DHT design. They reason that, unlike Shoup’s construction [44], the key generation in threshold BLS signature scheme does not mandate a trusted dealer, and unlike Gennaro et al.’s construction [21], the signing protocol in threshold BLS signature scheme does not require any interaction among peers or any zero-knowledge proofs. They also mention efficiency of the BLS signature scheme in terms of size and generation algorithm as compared to the other options and employ it to authenticate the communication between the quorums.

## 4.2 Distributed Key Generation—DKG

As a trusted party is not feasible in the P2P paradigm, the underlying robust DHT architecture also needs a complete distributed setup in the form of DKG to generate distributed signing keys. An  $(\eta, t)$ -DKG protocol allows a set of  $\eta$  nodes to construct a shared secret key  $sk$  such that its shares  $sk_i$  are distributed across the nodes and no coalition of fewer than  $t$  nodes may reconstruct the secret. In the discrete logarithm setting, there is also an associated public key  $PK$  and a set of public key shares  $\widehat{PK}$  in DKG for verification as required for threshold signatures.

For the robust DHT architecture, Young et al. use a DKG protocol [24] defined for use over the Internet. We continue to use threshold BLS signatures over this DKG setup in our privacy preserving enhancement.

## 4.3 Oblivious Transfer—OT

The first notion of oblivious transfers was introduced in 1981 by Rabin [37]. A 1-out-of-2 oblivious transfer (OT) [17] allows a chooser<sup>1</sup>  $p$  to decide between two messages held by a server<sup>2</sup>  $q$ . Moreover, OT protocols also guarantee that the server learns nothing, while the chooser obtains at most one of the messages. The concept may be generalized to 1-out-of- $\nu$  OT, where  $q$  holds  $\nu$  messages from which  $p$  may pick only one. In this work we will use this to obtain the relevant entry of the routing table from a quorum; the use of oblivious transfers ensures that the query remains secret, while at the same time spamming is prevented, since a malicious  $p$  is guaranteed to receive only a single entry.

We utilize an OT protocol by Naor and Pinkas [32, Protocol 3.1] as it fulfills all our needs; see Appendix A for an overview. The protocol provides 1-out-of- $\nu$  string OT, as we require. It is round optimal

<sup>1</sup>This is sometimes denoted “receiver” in the OT literature; we use the term “chooser” to avoid confusion with the overall receiver of message  $m$  in the surrounding DHT protocol.

<sup>2</sup>This is typically denoted “sender” in the OT literature; we use the term “server” to avoid confusion with the overall sender of message  $m$  in the surrounding DHT protocol.

and requires only one message per party (*OT-request* from  $p$  and *OT-response* from  $q$ ), except an *OT-setup* message that we may piggyback in the surrounding protocol. Moreover, it requires no zero-knowledge proofs, and also works in the elliptic curve cryptography (ECC) setting. The computation complexity of the protocol is dominated by the number of exponentiations; both server and chooser must on average perform two of these. In addition to the low computational costs, the overall communication amounts to roughly  $3\nu$  group elements.

The construction of Naor and Pinkas allows transfer of group elements; i.e., strings of approximately 256 bits in the ECC setting. This is not sufficient for an entire entry of a routing table. Rather than increasing the group size or performing multiple OTs, we simply let a peer  $q$  symmetrically (AES) encrypt each entry of the routing table using a random key. The encrypted table is then sent to peer  $p$  who uses an OT execution to obtain the AES key for the relevant entry from peer  $q$ .

For protocol RCPqp-I in Section 5.2, we will require a chooser  $p$  to run an OT with multiple members of the same quorum. We could reduce  $p$ 's computation by ensuring that all parallel OT instances are verbatim copies here. This would naturally require that the all servers use the same source of randomness for *OT-setup* and for AES keys. This can be achieved easily using a parameterized pseudorandom function (PRF):  $\phi(r, \cdot)$ . The private key  $r$  required for  $\phi$  can easily be agreed upon as part of a DKG execution, as it should be known to all quorum members. When the quorum executes an OT instance with chooser  $p$ , it may use  $p$ 's message itself as an input to PRF  $\phi$ . This PRF-based modification does not have any effect on the OT security proof as all parallel OT instances are verbatim copies.

**Other Possibilities** A natural question to ask is whether OT is really required, or whether another protocol could achieve the desired goal more efficiently. Although PIR protocols appear to be an alternative, they are not an acceptable alternative because they leak routing information. Further, computational PIR protocols have similar cost as the selected OT protocol [32]. For that matter, most non-trivial PIR is essentially OT as well.

Theoretically better OT protocols also exist, e.g. Lipmaa's OT protocol [27], which provides 1-out-of- $\nu$  OT with  $O(\log^2 \nu)$  multiplicative overhead on communication (of a single entry). For the proposed protocol by Naor and Pinkas, the overhead is linear which, in theory, is clearly worse. However, our approach is better in the present setting when we consider numbers from real-world DHTs. With more than million peers in a practical DHT, we will have  $\nu \approx 20$ . For  $\nu \approx 20$ ,  $\log^2 \nu \approx 20$  and linear communication without any hidden constant is quite acceptable. A generic 1-out-of-2 OT protocol of Peikert et al. [36] requires only two messages, and roughly five exponentiations per party. However, this is still more than the amortized cost of the 1-out-of- $\nu$  OT of Naor and Pinkas and we do not use it. While we cannot rule out the possibility of a more efficient protocol, it seems highly unlikely.

Finally, hiding the range values in routing table entries seems possible, but it is most likely infeasible in practice. Blake and Kolesnikov [8] provides a 1-out-of-2 conditional OT (COT) based on the greater-than relation. Their protocol has a blowup of a factor linear in the bitlength of the key. This blowup is needed in order to compute the greater-than relation. In addition to this, there are two critical issues that must be solved before COT can be used for hiding the range values: 1) the present work [8] requires a 1-out-of- $\nu$  conditional oblivious transfer 2) the protocols of [8] are only secure against semi-honest adversaries. Neither seems impossible to solve, but both appear to incur a significant blowup. Nevertheless, as no routing information is lost through range boundaries, we need not consider these.

## 5 Adding Query Privacy

Young et al. [52] present two robust communication protocols using quorums and threshold cryptography: RCP-I and RCP-II. As described in Section 3.2, both these protocols work in the general communication



architecture shown in Figure 2. They use threshold BLS signatures over the DKG architecture explained in sections 4.1 and 4.2. In this section, we provide query privacy to the above protocols using the OT primitive explained in Section 4.3 to define protocols RCPqp-I and RCPqp-II.

## 5.1 System Setup

We start our discussion by describing the setup required for our protocols. For clarity of description, we also briefly review routing tables ( $\mathcal{RT}$ ) in quorum-based DHTs.

**Initiation.** Before the system becomes functional, the initiator has to choose appropriate groups and other setup parameters for the BLS signature and OT protocols. Note that there are no trust assumptions required during this step, as these parameters can be selected from the well-known standards.

**Distributed Key Generation.** A DKG instance is executed, when a quorum gets formed in DHTs. At the end of an execution, each quorum  $Q_i$  is associated with a (distributed) public/private key pair  $(PK_{Q_i}, sk_{Q_i})$ . Note that only those quorums linked to  $Q_i$ , and not everyone in the network, need to know  $PK_{Q_i}$ . Further, every peer  $p \in Q_i$  possesses a private key share  $(sk_{Q_i})_p$  of  $sk_{Q_i}$ . Unlike the quorum public/private key pair of  $Q_i$  which must be known to all quorums to which  $Q_i$  is linked in the quorum topology, only the members of  $Q_i$  need to know the corresponding public key shares  $\widehat{PK}_{Q_i}$ . The private key  $r$  of PRF  $\phi(r, \cdot)$  required in RCPqp-I can easily be generated during this DKG execution.

**Routing Table Setup.** Without loss of generality, we assume a Chord-like DHT [47]. When a quorum gets formed in DHTs, it determines its neighbors and forms its routing table  $\mathcal{RT}$ . For a quorum  $Q_i$ , each entry of its routing table has the form  $\mathcal{RT}_{Q_j} = [Q_j, p, p', PK_{Q_j}, ts]$ . In this entry, peer  $p \in Q_j$  and peer  $p' \in Q_{j-1}$  where quorum  $Q_i$  links to quorum  $Q_j$  and  $Q_{j-1}$  in the quorum topology and  $p$  and  $p'$  are respectively located clockwise of all other peers in  $Q_j$  and  $Q_{j-1}$ .  $PK_{Q_j}$  is the quorum public key of  $Q_j$  generated using DKG, and  $ts$  is a time stamp for when this entry was created. Quorum  $Q_j$  is responsible for the identifier space between identities  $p$  and  $p'$ .  $\mathcal{RT}$  entries of  $Q_i$  are set such that the complete identifier space is covered by them.

## 5.2 Adding Query Privacy to RCP-I: RCPqp-I

Protocol RCP-I works deterministically. Here, we include a privacy preserving mechanism for queries in RCP-I using the OT protocol described in Section 4.3. The enhanced protocol (RCPqp-I) appears in Figure 3, which we outline as follows.

Assume that  $p \in Q_1$  is searching for a key and the target is a set of peers  $D \subseteq Q_\ell$ . Let the search path go through quorums  $Q_1, \dots, Q_\ell$ . Peer  $p$  begins by sending a request  $[p, p_{\text{addr}}, ts_1]$  to all peers in its quorum  $Q_1$ , where  $ts_1$  is a time stamp. Unlike the original RCP-I, the key corresponding to the intended destination of the message is not included here. Each honest peer  $q \in Q_1$  checks if  $p$ 's request follows the rule-set as described in Section 3.2. If there is no violation,  $q$  sends its signature share to  $p$ , who interpolates those shares to generate a signature  $S_1 = [p|p_{\text{addr}}|ts_1]_{sk_{Q_1}}$ . In each intermediate step ( $i = 2$  to  $\ell - 1$ ),  $p$  sends its most recent signature  $S_{i-1}$  and a new time stamp  $ts_i$  to each peer  $q \in Q_i$ . Since  $Q_i$  is linked to  $Q_{i-1}$  in the quorum topology, each peer  $q$  knows public key  $PK_{Q_{i-1}}$  to verify  $S_{i-1}$ . If  $S_{i-1}$  is verified and  $ts_i$  is valid, peer  $q$  sends back its signature share on  $[p|p_{\text{addr}}|ts_i]$ . Peer  $p$  collects the shares to form  $S_i$  and majority filters on the routing information for  $Q_{i+1}$ . If verification of  $S_i$  fails, peer  $p$  sends all shares back to every party in  $Q_i$ , who help  $p$  by filtering the invalid shares out. Finally, for  $Q_\ell$ ,  $p$  sends  $m$  along with  $S_{\ell-1}$  to peers in the target set  $D$  in  $Q_\ell$ .

<b>Initial Step:</b> $p \in \mathbf{Q}_1$ with Quorum $\mathbf{Q}_1$		
<b>peer <math>p</math></b>		<b>every peer <math>q \in \mathbf{Q}_1</math></b>
sends a request $[p p_{\text{addr}} ts_1]$	$\Rightarrow$	
	$\Leftarrow$	if the request is legitimate, reply with a signature share
<b>Intermediate Steps:</b> $p \in \mathbf{Q}_1$ with Quorum $\mathbf{Q}_i$ for $i = 2$ to $\ell - 1$		
<b>peer <math>p</math></b>		<b>every peer <math>q \in \mathbf{Q}_i</math></b>
interpolate $\mathcal{S}_{i-1} = [p p_{\text{addr}} ts_{i-1}]_{sk_{\mathbf{Q}_{i-1}}}$ using the received shares and send $\mathcal{S}_{i-1}$ and a new $ts_i$ . Request an <i>OT initiation</i>	$\Rightarrow$	
	$\Leftarrow$	verify $\mathcal{S}_{i-1}$ using $PK_{\mathbf{Q}_{i-1}}$ and validates $ts_i$ . If successful, send a signature share, an <i>OT-setup</i> message, the <i>ranges in <math>\mathcal{RT}</math> of <math>\mathbf{Q}_i</math></i> and the <i>entry-wise encrypted <math>\mathcal{RT}</math> of <math>\mathbf{Q}_i</math></i>
interpolate $\mathcal{S}_i = [p p_{\text{addr}} ts_i]_{sk_{\mathbf{Q}_i}}$ using the received shares and verify it using $PK_{\mathbf{Q}_i}$ . If invalid, sends all signature shares back. Send an <i>OT-request for the index corresponding to the searched key</i>	$\Rightarrow$	
	$\Leftarrow$	verify all shares using $\widehat{PK}_{\mathbf{Q}_i}$ and inform $p$ of valid shares. Send an <i>OT-response</i>
<i>Use the received OT-responses, if any, to determine the next quorum <math>\mathbf{Q}_{i+1}</math></i>		
<b>Final Step:</b> $p \in \mathbf{Q}_1$ with Quorum $\mathbf{Q}_\ell$		
<b>peer <math>p</math></b>		$D \subseteq \mathbf{Q}_\ell$
send $\mathcal{S}_{\ell-1}$ along with its request $m$	$\Rightarrow$	

Figure 3: RCPqp-I: RCP-I with Query Privacy

It still remains to see how  $\mathbf{Q}_i$  tells  $p$  the correct  $\mathbf{Q}_{i+1}$  as the next quorum without knowing the key being searched for. We accomplish this using the OT protocol. Along with  $\mathcal{S}_{i-1}$  and  $ts_i$ ,  $p$  also sends an *OT-initiation* request to every peer in  $q \in \mathbf{Q}_i$ . Peer  $q$  responds back with the entry-wise symmetrically encrypted (AES) routing table  $\mathcal{RT}_{\mathbf{Q}_i}$ , the *OT-setup* message, and the upper and lower bounds of ranges in  $\mathcal{RT}_{\mathbf{Q}_i}$ . Note that since all quorum members use the same randomness (due to the use of a PRF where everyone holds the private key), the messages from all honest parties will be the same. Peer  $p$  determines an index in  $\mathcal{RT}_{\mathbf{Q}_i}$  for the next quorum by searching for **key** in the received ranges and sends an *OT-request* for that index. Peer  $q$  then computes and sends the *OT-response*. Using this response, peer  $p$  obtains the symmetric key corresponding to the queried index and decrypts the appropriate entry in  $\mathcal{RT}_{\mathbf{Q}_i}$  to determine the next quorum  $\mathbf{Q}_{i+1}$ . Any wrongdoing by Byzantine peers in  $\mathcal{RT}_{\mathbf{Q}_i}$  range tables, encrypted  $\mathcal{RT}_{\mathbf{Q}_i}$  blocks and OT executions are taken care of by the majority action. As  $p$  knows  $\mathbf{Q}_2$  using its own routing table, there is no OT involved in the initial step.

Notice that it is also possible for peer  $p$  to use OT in the final step while communicating with the target set  $D$  in  $\mathbf{Q}_\ell$ , if the privacy application demands it. In that case, the target set  $D$  only knows that the queried key is one of its keys, but cannot determine the exact key.

<b>Initial Step:</b> $p \in \mathbf{Q}_1$ with Quorum $\mathbf{Q}_1$		
<b>peer <math>p</math></b>		<b>every peer <math>q \in \mathbf{Q}_1</math></b>
sends a request $[p p_{\text{addr}} ts]$	$\implies$	
	$\longleftarrow$	if the request is legitimate, reply with a signature share
verify and interpolate received shares to form $M_1 = [p p_{\text{addr}} ts_1]_{sk_{\mathbf{Q}_1}}$		
<b>Intermediate Steps:</b> $p \in \mathbf{Q}_1$ with Quorum $\mathbf{Q}_i$ for $i = 2$ to $\ell - 1$		
<b>peer <math>p</math></b>		<b>selected peer <math>q_i \in \mathbf{Q}_i</math></b>
select peer $q \in \mathbf{Q}_i$ uniformly at random without replacement. Send $M_{i-1}$ and request an <i>OT initiation</i>	$\implies$	
	$\longleftarrow$	For $j = i-1$ downto 2, verify $PK_{\mathbf{Q}_{j-1}}$ using $PK_{\mathbf{Q}_j}$ and verify $M_1$ using $PK_{\mathbf{Q}_1}$ . If successful, send $[PK_{\mathbf{Q}_{i-1}}]_{sk_{\mathbf{Q}_i}}$ , the <i>ranges in <math>\mathcal{RT}</math> of <math>\mathbf{Q}_i</math> and the entry-wise encrypted (signed) <math>\mathcal{RT}</math> of <math>\mathbf{Q}_i</math></i>
sends an <i>OT-request</i> for the index corresponding to the searched key	$\implies$	
	$\longleftarrow$	send an <i>OT-response</i> back
If $PK_{\mathbf{Q}_{i+1}}$ , $\mathcal{RT}_{\mathbf{Q}_{i+1}}$ (computed from the <i>OT-response</i> ) and $PK_{\mathbf{Q}_{i-1}}$ verifies, compute $M_i = [M_{i-1} [PK_{\mathbf{Q}_{i-1}}]_{sk_{\mathbf{Q}_i}}]$ and determine the next quorum $\mathbf{Q}_{i+1}$ from $\mathcal{RT}_{\mathbf{Q}_{i+1}}$ . Otherwise or if there is a timeout, choose $q'_i \in_R \mathbf{Q}_i$ and repeat		
<b>Final Step:</b> $p \in \mathbf{Q}_1$ with Quorum $\mathbf{Q}_\ell$		
<b>peer <math>p</math></b>		$D \subseteq \mathbf{Q}_\ell$
send $M_{\ell-1}$ along with its request $m$	$\implies$	

Figure 4: RCPqp-II: RCP-II with Query Privacy

The correctness of protocol RCPqp-I follows directly from that of protocol RCP-I and we refer the readers to [53] for a detailed proof. Although the encrypted routing tables (a few kilobytes in size) are sent in our privacy-preserving approach as compared to the individual routing table entires in RCPqp-I, it does not affect the message complexity of the protocol. The message complexity of protocol RCPqp-I remains exactly the same as protocol RCP-I, which is equal to  $O(\log^2 n)$ . We discuss the increase in computational cost and other systems matters in Section 6.

### 5.3 Adding Query Privacy to RCP-II: RCPqp-II

Protocol RCP-II utilizes signed routing table ( $\mathcal{RT}$ ) information and reduces the message complexity in protocol RCP-I by a linear factor (in expectation) using a uniformly random selection of peers in the quorums. Here, all  $\mathcal{RT}$  entries are signed separately by the quorum whenever  $\mathcal{RT}$ s are modified. In particular, every peer in the quorum, using their DKG private key shares, generates and sends signature shares, which are

then interpolated to obtain signed  $\mathcal{RT}$  entries. The OT setup and the OT protocol remain exactly the same as in RCPqp-I. The enhanced protocol (RCPqp-II) appears in Figure 4, which we outline as follows.

Initially, for simplicity, assume that peers act correctly. The initial step, where  $p$  communicates within its quorum,  $\mathbf{Q}_1$ , remains exactly the same. Each peer in  $\mathbf{Q}_1$  receives  $[p|p_{\text{addr}}|ts]$  from  $p$ . If the request does not violate the rule set, then peer  $p$  receives signature shares and computes  $M_1 = [p, p_{\text{addr}}, ts]_{sk_{\mathbf{Q}_1}}$ . Next,  $p$  knows the membership of  $\mathbf{Q}_2$  which belongs to its  $\mathcal{RT}$ , and selects a peer  $q_2 \in \mathbf{Q}_2$  uniformly at random without replacement. Peer  $p$  sends  $M_1$  to  $q_2$ . The correct  $q_2$  verifies  $M_1$  using  $PK_{\mathbf{Q}_1}$ , and replies with  $[PK_{\mathbf{Q}_1}]_{sk_{\mathbf{Q}_2}}$  and  $[\mathcal{RT}_{\mathbf{Q}_3}]_{sk_{\mathbf{Q}_2}}$ . Here,  $[PK_{\mathbf{Q}_j}]_{sk_{\mathbf{Q}_i}}$  denotes the quorum public key of  $\mathbf{Q}_j$  signed by quorum  $\mathbf{Q}_i$  as neighboring quorums know each others' public keys, and  $[\mathcal{RT}_{\mathbf{Q}_j}]_{sk_{\mathbf{Q}_i}}$  denotes the routing entry for  $\mathbf{Q}_j$  signed by  $\mathbf{Q}_i$ . Peer  $p$  verifies  $[PK_{\mathbf{Q}_1}]_{sk_{\mathbf{Q}_2}}$  and  $[\mathcal{RT}_{\mathbf{Q}_3}]_{sk_{\mathbf{Q}_2}}$ , and checks if the time stamp is valid. If so,  $p$  constructs  $M_2 = [M_1|[PK_{\mathbf{Q}_1}]_{sk_{\mathbf{Q}_2}}]$ . The idea is to allow some peer in  $\mathbf{Q}_3$  to verify  $PK_{\mathbf{Q}_1}$  and  $M_1$  using a signature chain. Further,  $p$  can check the response from some peer in  $\mathbf{Q}_3$  in the next step using  $PK_{\mathbf{Q}_3}$  included in  $\mathcal{RT}_{\mathbf{Q}_3}$ . This process repeats with minor changes for the remaining steps until  $p$  reaches the destination quorum  $\mathbf{Q}_\ell$ . If any peer does not respond in the amount of time predefined by the weak synchrony assumption [12] (as described in Section 3.1) or responds incorrectly, the protocol proceeds by choosing uniformly at random another peer in the quorum. Note that any attempt by a malicious peer to return incorrect information is detectable.

It still remains to see how the OT executions for key are performed such that a correct peer  $q_i$  in  $\mathbf{Q}_i$  can give routing information for  $\mathbf{Q}_{i+1}$  to peer  $p$ . For this, peer  $p$  sends an *OT initiation* to peer  $q_i$  along with  $M_{i-1}$ . Upon verification of the signature chain,  $q_i$  replies with  $[PK_{\mathbf{Q}_{i-1}}]_{sk_{\mathbf{Q}_i}}$ , ranges in  $\mathcal{RT}$  of  $\mathbf{Q}_i$ , entry-wise encrypted (signed)  $\mathcal{RT}$  of  $\mathbf{Q}_i$ , and the OT-setup message. Note that these encryptions are done locally at peers, and applied on both the  $\mathcal{RT}$  entires and signatures. Peer  $p$  then determines an index corresponding to the key it is searching for and sends an *OT-request* for that index. Peer  $q_i$  then computes and sends an *OT-response*. Using this response, peer  $p$  obtains the symmetric key corresponding to the queried index and decrypts the appropriate entry in  $\mathcal{RT}_{\mathbf{Q}_i}$ , checks the signature on the resulting plaintext, and thus determines the next quorum  $\mathbf{Q}_{i+1}$ .

Similar to RCPqp-I, if required, it is possible for peer  $p$  to use OT in the final step while communicating with the target set  $D$  in  $\mathbf{Q}_\ell$ . The correctness of the protocol follows directly from that of the original RCP-II protocol, and we refer the readers to [53] for a detailed proof. The message complexity of the enhanced protocol remains exactly the same as the original protocol, which is equal to  $O(\log n)$  in expectation. We discuss the increase in computational cost and other systems matters in Section 6.

## 6 Analysis and Discussion

As discussed in Section 5, our protocols do not increase the message complexity of their original counterparts RCP-I and RCP-II. In this section, we consider the increase in computation due to the query-privacy mechanism and find it to be nominal. We also analyze possible system-level attacks on our protocols.

### 6.1 Additions to Computational Costs

Query privacy does not come without some additional computation. However, for our choice of OT, this increase is insignificant as compared to the computations already done in the original RCP-I and RCP-II protocols.

In both the RCPqp-I and RCPqp-II protocols, a requesting peer  $p$  has to perform only two additional exponentiations at each privacy-preserving  $\mathcal{RT}$  entry retrieval, while a responding peer  $q_i$  in quorum  $\mathbf{Q}_i$  must perform one additional exponentiation. Peers in  $\mathbf{Q}_i$  also have to perform  $\nu$  exponentiations for an OT setup, where  $\nu$  is the size of  $\mathcal{RT}$ . However, they can be batch-computed and may also be reused in

$\nu$  requests. In terms of computation, our privacy-preserving mechanism remains exactly the same in both protocols, RCPqp-I and RCPqp-II. This results from a peer  $p$  running the same instance of OT with all peers in the quorum in RCPqp-I with the help of the PRF-based technique discussed in Section 4.3.

Timing values computed using the pairing-based cryptography (PBC) library [28] indicate that one exponentiation takes around 1 ms on a desktop machine. Given that the communication time for the original RCP-I and RCP-II protocols is greater than 3 seconds (refer to Young et al. [52] for a detailed discussion), the cost of these exponentiations is insignificant. In terms of system load, a DKG execution in RCP-I and RCP-II on average requires 2 CPU seconds, and a threshold signature generation and verification takes about 6 CPU ms. Therefore, our OT executions do not increase the system load by any significant fraction. Note that the OT protocol also involves a few group multiplications, PRF executions, symmetric encryptions and hashes. Their computations take only a few  $\mu s$ , so we ignore these computational costs in our discussion.

## 6.2 System-level Attacks on Query Privacy

Although OT hides the queried key completely in the cryptographic sense, there can be system-level attacks that leak some information about the key.

A range estimation attack defined by Wang, Mittal and Borisov [50] that reduces privacy provided in NISAN [35] could be applied to our RCPqp-I protocol. This attack is based on the fact that the Chord-like DHT ring is directed and the requesting peer  $p$  will not query a quorum succeeding the queried key except in the first iteration. Therefore, an adversary that can observe the peer  $p$  contacting a sequence of quorums can put them together into a sequence to narrow down on the target range that peer  $p$  may reach. In this attack, the range only extends from the last contacted quorum having an adversarial peer to the largest jump possible at the end of first iteration. For NISAN, Wang et al. show that if at least 20% of nodes are under the adversary's control, the adversary may obtain a significant amount of information about the queried key. As indicated in Section 3.1, we consider the percentage of peers under the control of a single adversary to be around 10%. Therefore, although this range estimation attack is possible, it is not particularly effective in our DHT setting. On the other hand, the curious peers in the intermediate quorums only see requests approved by one of their neighbors. This, along with the security provided by the OT protocol, ensures that nothing is revealed about the queried key to the curious intermediate quorums.

As only an expected constant number of peers are contacted per intermediate quorum in our RCPqp-II protocol, the range estimation attack by Wang et al. [50] is far less effective. However, query privacy for our RCPqp-II protocol is slightly weaker in terms of the above mentioned curious observer attack. This is a direct consequence of the use of a signature chain to authorize a request from a peer  $p$ : assume a peer  $q_i$  from an intermediate quorum  $Q_i$ . Although  $q_i$  may not be able to determine quorums from the public keys in a chain, the length of the chain itself might give peer  $q_i$  some information about possible key values. This results from a property of Chord-like DHTs: *generally* each step brings a requester exponentially closer to its destination. As an example, a shorter signature chain indicates that a destination quorum is probably situated away from  $Q_i$  in the key (or identifier) space, while a length nearly equal to  $\log n$  indicates that the destination quorum is probably nearby. This is, however, a weak heuristic attack as path lengths of DHT requests may vary significantly. Further, it is possible to mislead such a curious adversary by adding a few fake signatures at the end of the chain. The requesting peer  $p$  has to have this done by its quorum  $Q_1$ .

## 6.3 Crawling Attacks towards Spam Prevention

As discussed in Section 2, usage of the iterative routing approach significantly improves robustness against spamming attacks, since a spamming peer has to perform an equal amount of work as the rest of the system. Young et al. [52] add further protections against spamming in RCP-I and RCP-II by not allowing the adversarial peer to gather a large amount of routing information. They add the queried keys to requests.

As a result, an execution of RCP-I or RCP-II leads to the requester  $p$  gaining information only about the  $\ell$  quorums in its path. We concentrate on query privacy in this work and enforce that the queried **key** should remain completely oblivious to every intermediate quorum  $Q_i$  for  $i \in [1, \ell - 1]$ . This may lead to attacks, where the adversary peer  $p$  obtains more routing information; we call these attacks *crawling attacks*.

In our RCPqp-I protocol, a malicious peer  $p$  may try to obtain the entire  $\mathcal{RT}$  of  $Q_i$  by querying for different **keys** (or  $\mathcal{RT}$  indices) to different peers in  $Q_i$ . As a result, the adversary peer  $p$  can acquire more information than allowed by the rule set. It is possible to thwart this supposed attack completely by adding one communication round: here,  $p$  also has to get its *OT-request* message (which is the same for all peers in  $Q_i$ ) signed from  $Q_i$  in the exact same way as its authorization request  $[p|p_{\text{addr}}|ts_i]$ . This ensures that  $p$  can query the quorum for only one **key** (specifically, one index in  $\mathcal{RT}$ ), and query privacy of the **key** also remains unaffected. This additional one round does not change the message complexity of the protocol. We do not include this defense mechanism in the protocol described in Figure 3, as repercussions of this attack, if any, may vary from system to system.

In our RCPqp-II protocol, similar crawling is possible. The adversary peer  $p$  may query different peers in quorum  $Q_i$  for different indices to obtain the complete  $\mathcal{RT}$  for  $Q_i$ . However, unlike in RCPqp-I, a malicious peer  $p$  has to increase its effort linearly to obtain the complete  $\mathcal{RT}$  of  $Q_i$  in RCPqp-II and crawling is not an effective attack for the malicious peer  $p$ .

In both protocols, it is possible for a malicious peer  $p$  to alter the queried **key** while shifting from one quorum to the next, as there is no link between signed authorizations and the queried **keys** for privacy reasons. This may, however, lead to a peer  $p$  gaining more knowledge as it can continuously modify its **key** to traverse as much of the DHT as possible. This is an even weaker crawling attack than the one mentioned above, as the adversary has to perform a significant amount of work to gain any information.

Notice that any information gained by the adversary in the above active attacks is still substantially smaller than information effortlessly available to it when PIR or trivial PIR are used. Finally, it may be possible to stop the adversary  $p$  from gaining any additional information without revealing its **key** using computationally and communicationally demanding cryptographic primitives such as zero-knowledge proofs or conditional OT [15]. However, we find that their inclusions are not essential, and may be even impractical, for DHT-based systems.

## 7 Conclusion

In this paper, we have introduced the concept of query privacy in the robust DHT architecture. We have enhanced two existing robust communication protocols (RCP-I and RCP-II) over DHTs to preserve the privacy of keys in DHT queries using an OT protocol. We reviewed the OT literature and chose a theoretically non-optimal but practically efficient (in terms of use over DHTs in practice) OT scheme. Using this, we built two protocols (RCPqp-I and RCPqp-II), which obtain query privacy without any significant increase in computation costs and message complexity in practice. Our privacy-preserving mechanism does not change the underlying protocols' utility or efficacy in any way, and is also applicable to other DHT communication architectures.

## References

- [1] Electronic Privacy Information Center (EPIC). <https://www.eff.org/>, 1990. Accessed March 2012.
- [2] Electronic Frontier Foundation (EFF). <http://epic.org/>, 1994. Accessed March 2012.

- [3] J. Aspnes, N. Rustagi, and J. Saia. Worm versus alert: Who Wins in a Battle for Control of a Large-Scale Network? In *Intl. Conference on Principles of Distributed Systems*, pages 443–456, 2007.
- [4] B. Awerbuch and C. Scheideler. Group Spreading: A Protocol for Provably Secure Distributed Name Service. In *ICALP*, pages 183–195, 2004.
- [5] B. Awerbuch and C. Scheideler. Robust Random Number Generation for Peer-to-Peer Systems. In *OPODIS*, pages 275–289, 2006.
- [6] B. Awerbuch and C. Scheideler. Towards a Scalable and Robust DHT. In *Symposium on Parallelism in Algorithms and Architectures (SPAA)*, pages 318–327, 2006.
- [7] B. Awerbuch and C. Scheideler. Towards Scalable and Robust Overlay Networks. In *Intl. Workshop on Peer-to-Peer Systems*, 2007.
- [8] I. F. Blake and V. Kolesnikov. One-round secure comparison of integers. *Journal of Mathematical Cryptography*, 3(1), may 2009.
- [9] A. Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In *IACR PKC*, pages 31–46, 2003.
- [10] D. Boneh. The Decision Diffie-Hellman Problem. In *ANTS*, pages 48–63, 1998.
- [11] D. Boneh, B. Lynn, and H. Shacham. Short Signatures from the Weil Pairing. In *Advances in Cryptology—ASIACRYPT’01*, pages 514–532, 2001.
- [12] M. Castro and B. Liskov. Practical Byzantine Fault Tolerance and Proactive Recovery. *ACM Trans. Computer Systems*, 20(4):398–461, 2002.
- [13] B. Chor, E. Kushilevitz, O. Goldreich, and M. Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.
- [14] Y. G. Desmedt. Threshold cryptography. *European Transactions on Telecommunications*, 5(4), 1994.
- [15] G. Di Crescenzo, R. Ostrovsky, and S. Rajagopalan. Conditional oblivious transfer and timed-release encryption. In *EUROCRYPT*, pages 74–89, 1999.
- [16] R. Dingledine, N. Mathewson, and P. F. Syverson. Tor: The second-generation onion router. In *USENIX Security Symposium*, pages 303–320, 2004.
- [17] S. Even, O. Goldreich, and A. Lempel. A randomized protocol for signing contracts. *Commun. ACM*, 28:637–647, June 1985.
- [18] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, and T. Anderson. Profiling a Million User DHT. In *IMC*, pages 129 – 134, 2007.
- [19] A. Fiat, J. Saia, and M. Young. Making Chord Robust to Byzantine Attacks. In *European Symp. on Algorithms*, pages 803–814, 2005.
- [20] R. Geambasu, T. Kohno, A. A. Levy, and H. M. Levy. Vanish: Increasing Data Privacy with Self-Destructing Data. In *USENIX Security Symposium*, pages 299–315, 2009.
- [21] R. Gennaro, S. Jarecki, H. Krawczyk, and T. Rabin. Robust Threshold DSS Signatures. In *Advances in Cryptology - EUROCRYPT*, pages 354–371, 1996.

- [22] K. Hildrum and J. Kubiawicz. Asymptotically Efficient Approaches to Fault-Tolerance in Peer-to-peer Networks. In *DISC*, pages 321–336, 2004.
- [23] H. Johansen, A. Allavena, and R. van Renesse. Fireflies: Scalable Support for Intrusion-Tolerant Network Overlays. In *EuroSys Conference*, pages 3–13, 2006.
- [24] A. Kate and I. Goldberg. Distributed Key Generation for the Internet. In *IEEE ICDCS*, pages 119–128, 2009.
- [25] J. Liang and R. Kumar. Pollution in P2P file sharing systems. In *IEEE Intl. Conf. on Computer Communications (INFOCOM)*, 2005.
- [26] J. Liang, N. Naumov, and K. W. Ross. The Index Poisoning Attack in P2P File Sharing Systems. In *IEEE Intl. Conf. on Computer Communications (INFOCOM)*, pages 1–12, 2006.
- [27] H. Lipmaa. An oblivious transfer protocol with log-squared communication. In *ISC*, pages 314–328, 2005.
- [28] B. Lynn. The Pairing-Based Cryptography (PBC) Library. <http://crypto.stanford.edu/pbc/>, 2006. Accessed March 2012.
- [29] J. McLachlan, A. Tran, N. Hopper, and Y. Kim. Scalable onion routing with Torsk. In *ACM CCS*, pages 590–599, 2009.
- [30] P. Mittal and N. Borisov. ShadowWalker: Peer-to-peer Anonymous Communication using Redundant Structured Topologies. In *ACM CCS*, pages 161–172, 2009.
- [31] A. Nambiar and M. Wright. Salsa: A Structured Approach to Large-Scale Anonymity. In *ACM CCS*, pages 17–26, 2006.
- [32] M. Naor and B. Pinkas. Efficient oblivious transfer protocols. In *SODA*, pages 448–457, 2001.
- [33] M. Naor and U. Wieder. A Simple Fault Tolerant Distributed Hash Table. In *Intl. Workshop on Peer-to-Peer Sys.*, pages 88–97, 2003.
- [34] R. Narendula, T. G. Papaioannou, Z. Miklos, and K. Aberer. Tunable Privacy for Access Controlled Data in Peer-to-Peer Systems. In *International Teletraffic Congress (ITC 22)*, 2010.
- [35] A. Panchenko, S. Richter, and A. Rache. NISAN: network information service for anonymization networks. In *ACM CCS*, pages 141–150, 2009.
- [36] C. Peikert, V. Vaikuntanathan, and B. Waters. A Framework for Efficient and Composable Oblivious Transfer. In *CRYPTO*, pages 554–571, 2008.
- [37] M. Rabin. How to exchange secrets with oblivious transfer. Technical Report TR-81, Aiken Computation Lab, Harvard University, Cambridge, MA, 1981.
- [38] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Schenker. A Scalable Content-Addressable Network. In *SIGCOMM*, pages 161–172, 2001.
- [39] R. Rodrigues and B. Liskov. Rosebud: A Scalable Byzantine-Fault-Tolerant Storage Architecture. Technical Report TR/932, MIT LCS, December 2003.
- [40] R. Rodrigues, B. Liskov, K. Chen, M. Liskov, and D. M. Schultz. Automatic reconfiguration for large-scale reliable storage systems. *IEEE TDSC*, 99:1, 2010.



- [41] A. I. T. Rowstron and P. Druschel. Pastry: Scalable, Decentralized Object Location, and Routing for Large-Scale Peer-to-Peer Systems. In *Middleware*, pages 329–350, 2001.
- [42] J. Saia and M. Young. Reducing Communication Costs in Robust Peer-to-Peer Networks. *Inform. Process. Lett.*, 106(4):152–158, 2008.
- [43] D. A. Schultz, B. Liskov, and M. Liskov. MPSS: Mobile Proactive Secret Sharing. *ACM Trans. Inf. Syst. Secur.*, 13(4):34, 2010.
- [44] V. Shoup. Practical Threshold Signatures. In *Advances in Cryptology - EUROCRYPT*, pages 207–220, 2000.
- [45] E. Sit and R. Morris. Security Considerations for Peer-to-Peer Distributed Hash Tables. In *Intl. Workshop on Peer-to-Peer Systems*, pages 261–269, 2002.
- [46] M. Steiner, T. En-Najjary, and E. W. Biersack. A Global View of KAD. In *IMC*, pages 117 – 122, 2007.
- [47] I. Stoica, R. Morris, D. Karger, M. F. Kaashoek, and H. Balakrishnan. Chord: A Scalable Peer-to-peer Lookup Service for Internet Applications. In *ACM SIGCOMM Conference*, pages 149–160, 2001.
- [48] V. Pappas and D. Massey and A. Terzis and L. Zhang. A Comparative Study of the DNS Design with DHT-Based Alternatives. In *IEEE Intl. Conference on Computer Communications*, pages 1–13, 2006.
- [49] D. S. Wallach. A Survey of Peer-to-Peer Security Issues. In *Intl. Conf. on Software Security: Theories and System*, pages 253–258, 2002.
- [50] Q. Wang, P. Mittal, and N. Borisov. In search of an anonymous and secure lookup: attacks on structured peer-to-peer anonymous communication systems. In *ACM CCS*, pages 308–318, 2010.
- [51] S. Wolchok, O. S. Hofmann, N. Heninger, E. W. Felten, J. A. Halderman, C. J. Rossbach, B. Waters, and E. Witchel. Defeating Vanish with Low-Cost Sybil Attacks Against Large DHTs. In *NDSS*, 2010.
- [52] M. Young, A. Kate, I. Goldberg, and M. Karsten. Practical Robust Communication in DHTs Tolerating a Byzantine Adversary. In *IEEE ICDCS*, pages 263–272, 2010.
- [53] M. Young, A. Kate, I. Goldberg, and M. Karsten. Towards Practical Communication in Byzantine-Resistant DHTs. To appear in *IEEE/ACM Transactions on Networking (ToN)*, 2012.
- [54] B. Y. Zhao, L. Huang, J. Stribling, S. C. Rhea, A. D. Joseph, and J. Kubiatowicz. Tapestry: a resilient global-scale overlay for service deployment. *IEEE Journal on Selected Areas in Communications*, 22(1):41–53, 2004.

## A The Oblivious Transfer Protocol

In this appendix, we provide an overview of the 1-out-of- $\nu$  OT protocol of Naor and Pinkas [32]. Security of the construction is based on DDH in a group  $G$  of prime order  $|G|$ . The proof of security uses the random oracle model, i.e. the protocol uses a cryptographic hash function,  $H$ , which is then replaced by the random oracle in the proof. Recall that the goal is for the server,  $q$ , to offer  $\nu$  strings,  $S_1, \dots, S_\nu$ , and for chooser  $p$  to obtain the desired one,  $S_\rho$ , and nothing else. The basic idea of the [32] OT protocol is to let  $p$  provide encryption keys  $PK_i$  for  $1 \leq i \leq \nu$ ; these are constructed such that  $p$  can know at most one of the decryption keys. The server then supplies  $p$  with encryptions of each  $S_i$  under  $PK_i$ . Details are included in a protocol flow below in Figure 5. we now elaborate on the intuition behind the three messages:

<b>Setup</b> (for $\nu$ invocations)	
<b>peer <math>p</math></b>	<b>peer <math>q</math></b> Pick $r \in \mathbb{Z}_{ G }$ uniformly at random and compute $\alpha = g^r$ ; for $1 < i \leq \nu$ pick $C_i$ uniformly at random in $G$ and compute $C_i^r$ . Send $\alpha$ and $C_2, \dots, C_\nu$ to $p$ .
<b>Online</b> (single invocation)	
<b>peer <math>p</math> requesting <math>S_\rho</math></b> Pick $k \in \mathbb{Z}_{ G }$ uniformly at random and compute $PK_\rho = g^k$ . If $\rho \neq 1$ compute $PK_1 = C_\rho / PK_\rho$ . Send $PK_1$ to $q$ .	<b>peer <math>q</math> holding <math>S_1, \dots, S_\nu</math></b> Compute $PK_1^r$ ; then for $1 < i \leq \nu$ compute $PK_i^r = C_i^r / PK_1^r$ . Pick a random string $R$ and for $1 \leq i \leq \nu$ compute an encryption of $S_i$ , $H(PK_i^r, R, i) \oplus S_i$ ; send all $\nu$ encryptions to $p$ along with $R$ .
Compute first $PK_\rho^r = \alpha^k$ and then $H(PK_\rho^r, R, \rho)$ ; use this to decrypt the $\rho$ th encryption and output the plaintext, $S_\rho$ .	

Figure 5: The 1-out-of- $\nu$  OT protocol of Naor and Pinkas

1. **OT-setup:**  $q$  picks a random DL instance,  $\alpha = g^r$  and sends this to  $p$ . Moreover, the parties agree on  $\nu - 1$  random group elements,  $C_2, \dots, C_\nu$ . It is crucial that  $p$  does not know their DL, hence they are picked by  $q$  (who is allowed but not required to know the DL).
2. **OT-request:**  $p$  will supplies  $PK_1$  to  $q$ ; for  $1 < i \leq \nu$   $PK_i$  is implicitly set to  $C_i / PK_1$ .  $p$  constructs  $PK_1$  such that  $PK_\rho$  has known DL; however, if  $p$  could find the DL of any other key,  $PK_i$ ,  $p$  could solve a DDH problem in  $G$ .
3. **OT-response:**  $q$  computes  $PK_i^r = C_i^r / PK_1^r$  for all  $i$ . Since the  $C_i^r$  may be precomputed this requires only a single exponentiation and  $\nu - 1$  multiplications.  $q$  then picks a uniformly random  $\ell$ -bit string,  $R$ , where  $\ell$  is chosen large enough (e.g. 200 bits) to ensure that  $R$  will be distinct. Finally, for each  $1 \leq i \leq \nu$   $q$  computes an encryption of  $S_i$  as  $E_i = H(PK_i^r, R, i) \oplus S_i$ .  $R$  and the  $E_i$  are sent to  $p$ , who computes first  $PK_\rho^r = \alpha^k$  and then decrypts to obtain  $S_\rho = E_\rho \oplus H(PK_\rho^r, R, \rho)$ .

For more details along with the proof in the random oracle model, see [32]. As noted,  $\alpha$  and the  $C_i^r$  may be preprocessed during periods of low computational load. Moreover, the values may be used in multiple instances of the OT protocol. “Refreshing”  $r$  (i.e.  $\alpha$  and the  $C_i^r$ ) every  $\nu$  execution provides an amortized complexity of two exponentiations per party per OT invocation. The setup message consists of  $\nu$  group elements, while the OT-request contains only a single one. Finally, the reply consists of  $\nu$   $E_i$ s plus  $R$ ; though strictly speaking, these are not group elements, they may be viewed as such for the complexity analysis. Hence, overall communications is  $2\nu + 2$  group elements.

We remark that since we are transferring AES keys using OT, we could also directly use (some digest of)  $PK_i^r, R, i$  as the AES key. However, such ad hoc optimizations may easily introduce subtle flaws. The security proof of Naor and Pinkas may easily be invalidated by even a minor optimization, hence, as the gains are marginal, we prefer the original OT protocol to any ad hoc optimization.